IO

Yulin XIE

1.File

Some basic manipulations about `File`:

```
File test1 = new File("c:\\WINDOWS\\");
System.out.println("path: " + test1.getAbsolutePath());
System.out.println("existence: " + test1.exists());
System.out.println("directory: " + test1.isDirectory());
System.out.println("file: " + test1.isFile());
System.out.println("length: " + test1.length());
System.out.println("last modify time: " + new Date((long)test1.lastModified()));
test1.renameTo(new File("texe.exe"));
System.out.println("path: " + test1.getAbsolutePath());
System.out.println("father: " + test1.getParentFile());
```

2.Find specific file in directory

Find the maximum and minimum size file in a directory and print their sizes & names

- no sub-directory:

```
        System.out.println(Arrays.toString(test1.list()));
        File[]fs= test1.listFiles();
        long maxValue = Integer.MIN_VALUE;
        File maxName = null;
        long minValue = Integer.MAX_VALUE;
        File minName = null;
        for(File f: fs){
            if(f.length()>maxValue && f.length()!=0)
            {
                maxValue = f.length();
                maxName = f;
            }
            if(f.length()<minValue && f.length()!=0)
            {
                minValue = f.length();
                minName = f;
            }
        }
        assert maxName != null;
        System.out.println("Max: " + maxName.getAbsolutePath() + " " + maxValue);
        assert minName != null;
        System.out.println("Min: " + minName.getAbsolutePath() + " " + minValue);
```

- sub-directory:
  For this, w e do it in a recursive w ay:

```
    private static long maxValue1 = Integer.MIN_VALUE;
    private static File maxName1 = null;
    private static long minValue1 = Integer.MAX_VALUE;
    private static File minName1 = null;

    public static void find(File file){
            if(file.isDirectory()){
                File[] temp = file.listFiles();
                if(temp!=null)
```

```
                for(File f:temp)
                    find(f);
            }
            if(file.isFile()){
                if(file.length()>maxValue1 && file.length()!=0)
                {
                    maxValue1 = file.length();
                    maxName1 = file;
                }
                if(file.length()<minValue1 && file.length()!=0)
                {
                    minValue1 = file.length();
                    minName1 = file;
                }
            }
    }
    ...
    File test1 = new File("c:\\WINDOWS\\");
    find(test1);
    System.out.println("Max: " + maxName1.getAbsolutePath() + " " + maxValue1);
    System.out.println("Min: " + minName1.getAbsolutePath() + " " + minValue1);
```

3.Byte Stream

Pay attention:
OutputStream是字节输出流，同时也是抽象类，只提供方法声明，不提供方法的具体实现。
FileOutputStream 是OutputStream子类，以FileOutputStream 为例向文件写出数据

注: 如果文件d:/lol2.txt不存在，写出操作会自动创建该文件。
但是如果是文件 d:/xyz/lol2.txt，而目录xyz又不存在，会抛出异常
那么怎么自动创建xyz目录？:

```
package stream;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

public class TestStream {

    public static void main(String[] args) {
        try {
            File f = new File("d:/xyz/abc/def/lol2.txt");
            //因为默认情况下，文件系统中不存在 d:\xyz\abc\def，所以输出会失败
            //首先获取文件所在的目录
            File dir = f.getParentFile();
            //如果该目录不存在，则创建该目录
            if(!dir.exists()){
//              dir.mkdir(); //使用mkdir会抛出异常，因为该目录的父目录也不存在
                dir.mkdirs(); //使用mkdirs则会把不存在的目录都创建好
            }
            byte data[] = { 88, 89 };
            FileOutputStream fos = new FileOutputStream(f);
            fos.write(data);
            fos.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

4.Tear down file into several parts

Essence: read the file from hard disk into memory then divide it into w hatever w e w ant.

5.Close Stream in the right way

- primitive way:
  we close all the FileInputStream/FileOutputStream in the `finally` block to avoid forgetting to close them in `try` / `catch`

```
static public void read(File f){
        FileInputStream fis = null;//make sure that it is here insdead of in the 'try' block
        try{
            fis = new FileInputStream(f);//initialize here
            byte[] data = new byte[(int) f.length()];
            fis.read(data);//read data like this
            for(byte b:data)
                System.out.println(b);
        }catch (IOException e){
            e.printStackTrace();
        }finally {
            if(fis!=null){
                try { //need a 'try' block to enable the close manipulation
                    fis.close(); //we close the FileInputStream here
                }catch (IOException e){
                    e.printStackTrace();
                }
            }
        }
```

- avanced way(try-with-resources):
  we do everything in the `try(...)` block, so called **AutoCloseable** :

```
static public void read_avanced(File f){
        //把流定义在try()里,try,catch或者finally结束的时候, 会自动关闭
        try(FileInputStream fis = new FileInputStream(f)){
            byte[] data = new byte[(int) f.length()];
            fis.read(data);
            for(byte b:data)
                System.out.println(b);
        }catch (Exception e){
            e.printStackTrace();
        }
    }
```

未完待续....